Term Project

Semih Barutcu

2/17/2020

```
library(pacman)
p_load(dplyr, tidyverse, lubridate, Amelia, vtable, tictoc, rpart, rpart.plot, C50, ROCR,
       caret, randomForest, tictoc, ranger, class, gmodels, naivebayes)
```

In this project, Lending Club accepted loan data was studied. Loan status was response variable and I worked on both 84 remaining variable and selected 10 remaining variable. Machine Learning algorithms implementations and results can be seen below step by step.

Step 1 – Reading the data

```
LendingClub  <- read_csv("accepted_2007_to_2018Q4.csv") %>% mutate_if(is.character, as.factor)
```

I eliminated some variables because they are identifier variables train data which possibly overfit. Also, some of variables are not selected because of impractical to use and including excessive NA values.

```
#Defining year from issue_d and make it integer
LendingClub$year <- str_sub(LendingClub$issue_d, start=-4) %>% as.integer(LendingClub$year)
```

```
## Warning: Unknown or uninitialised column: `year`.
```

```
LendingClub_2012to2014 <- LendingClub %>%
  filter(between(year,2012,2014)) %>%
  select(-id, -member_id, -emp_title, -issue_d, -url, -desc, -zip_code, -title,
         -earliest_cr_line, -last_pymnt_d, -last_credit_pull_d)

LendingClub_2012to2014 <- LendingClub_2012to2014[, colMeans(is.na(LendingClub_2012to2014)) < 0.2]


LendingClub_2012to2014v2 <- LendingClub %>%
  filter(between(year,2012,2014)) %>%
  select(loan_status, funded_amnt, emp_length, annual_inc, last_pymnt_amnt, mort_acc,
         int_rate, mo_sin_old_rev_tl_op, avg_cur_bal, acc_open_past_24mths, num_bc_sats)
```

The dataset, which had 11 total variable, was named as v2 at the end of LendingClub, train and test data frames. These variables were selected according to ease of use them and I focused mostly numeric values. Table of variables can be seen below. There are 376,516 rows.

```
vtable(LendingClub_2012to2014v2, out = 'return')
```

```
##                   Name   Class
## 1          loan_status  factor
## 2          funded_amnt numeric
## 3           emp_length  factor
## 4           annual_inc numeric
## 5      last_pymnt_amnt numeric
```

```
## 6              mort_acc numeric
## 7              int_rate numeric
## 8  mo_sin_old_rev_tl_op numeric
## 9           avg_cur_bal numeric
## 10 acc_open_past_24mths numeric
## 11          num_bc_sats numeric
##
## 1  'Charged Off' 'Current' 'Default' 'Does not meet the credit policy. Status:Charged Off' 'Does not
## 2
## 3                                                                                           '< 1 yea
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
```

```r
nrow(LendingClub_2012to2014v2)
```

```
## [1] 423810
```

Step 2 – Exploring and preparing the data

From loan_status table, we can see that 3 results were observed at most. I filtered the data just for these options to get more accurate results.

On the first graph, loan status were depicted according to count of funded amounts and faceted by term. 36 months loan users had a right skewed distribution while 60 months users had an uneven distribution.

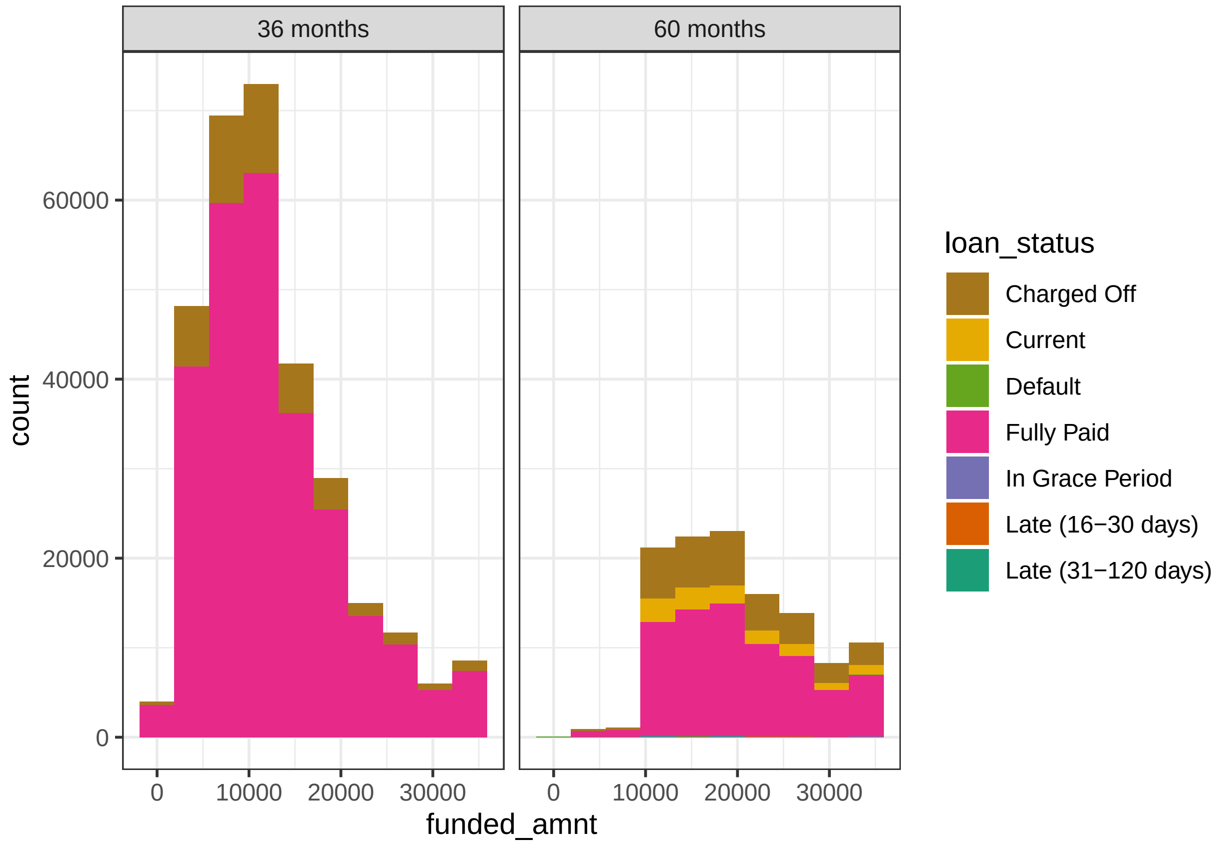
On the first graph, loan status were depicted according to count of interest rate and faceted by term. 36 months loan users had a right skewed distribution again while 60 months users had normal distribution.

We can see that 60 months term loan users had higher rate of charged off from table and graphs.

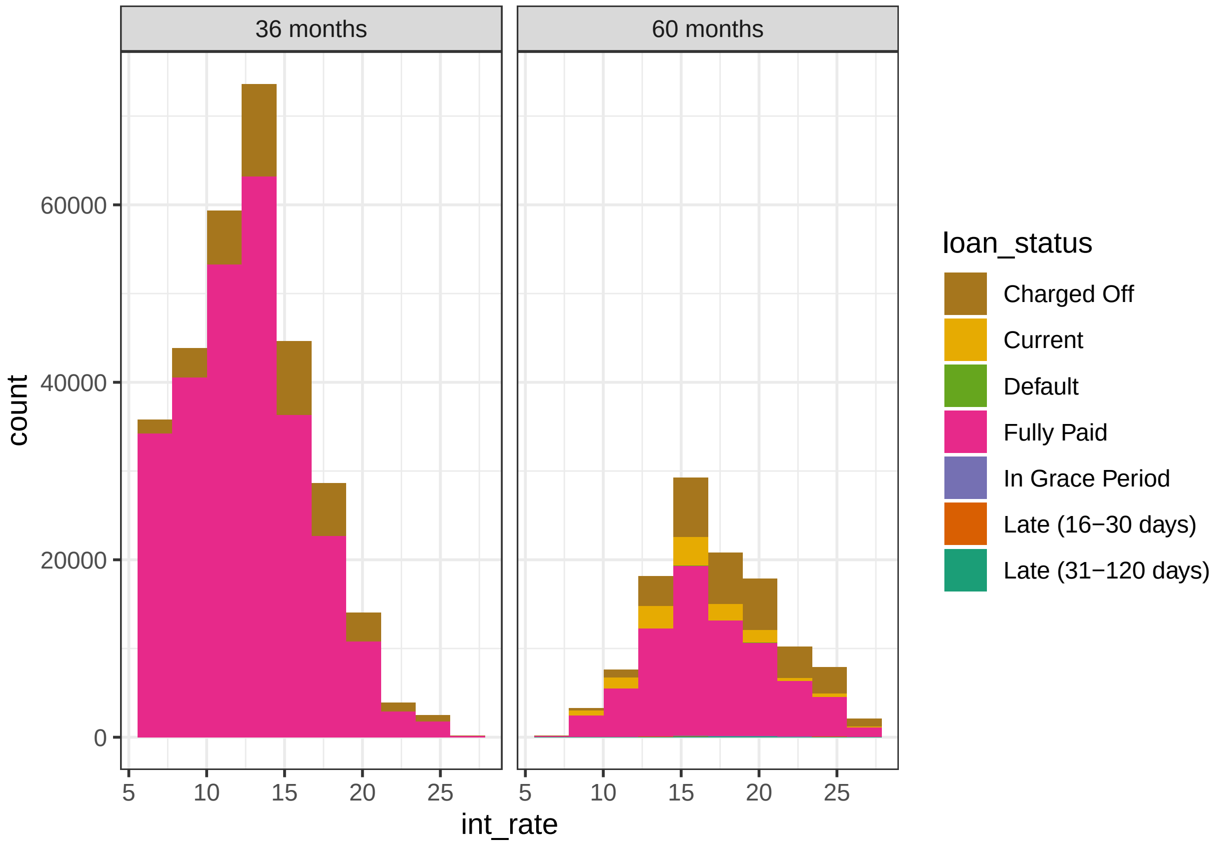
```r
table(LendingClub_2012to2014$loan_status, LendingClub_2012to2014$term)
```

```
##
##                                                         36 months 60 months
##   Charged Off                                               40596     30233
##   Current                                                       0     11925
##   Default                                                       0         1
##   Does not meet the credit policy. Status:Charged Off           0         0
##   Does not meet the credit policy. Status:Fully Paid            0         0
##   Fully Paid                                               265866     74578
##   In Grace Period                                               0       201
##   Late (16-30 days)                                             0        73
##   Late (31-120 days)                                            0       337
```

```r
LendingClub_2012to2014 %>%
  ggplot(aes(funded_amnt, fill = loan_status)) +
  geom_histogram(bins = 10) +
  scale_fill_brewer(palette = "Dark2", direction = -1) +
  facet_wrap(~term) +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```




```
LendingClub_2012to2014 %>%
  ggplot(aes(int_rate, fill = loan_status)) +
  geom_histogram(bins = 10) +
  scale_fill_brewer(palette = "Dark2", direction = -1) +
  facet_wrap(~term) +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```




```
LendingClub_2012to2014 <- LendingClub_2012to2014 %>%
  filter(loan_status == "Charged Off" | loan_status == "Current" | loan_status == "Fully Paid") %>% na.o

LendingClub_2012to2014v2 <- LendingClub_2012to2014v2 %>%
  filter(loan_status == "Charged Off" | loan_status == "Current" | loan_status == "Fully Paid") %>% na.o
```

Train and test data were created below with a 10000 and 2500 samples from the main data which resulted similar proportion of distribution with the main data. After all my work has done, I set a seed to interpret final results exactly.

```
set.seed(123)
idx1 <- sample(nrow(LendingClub_2012to2014v2), 10000)
idx2 <- sample(nrow(LendingClub_2012to2014v2), 2500)

idx <- sample(nrow(LendingClub_2012to2014v2), round(0.75*nrow(LendingClub_2012to2014v2)))

train <- LendingClub_2012to2014[idx1,]
```

```
## Warning: The `i` argument of ``[.tbl_df`()` must lie in [0, rows] if positive, as of tibble 3.0.0.
## Use `NA_integer_` as row index to obtain a row full of `NA` values.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
test <- LendingClub_2012to2014[idx2,]

trainv2 <- LendingClub_2012to2014v2[idx1,]
testv2 <- LendingClub_2012to2014v2[idx2,]
```

Levels of factor variable are reduced to 3 different types for loan status below.

```
levels(trainv2$loan_status)
```

```
## [1] "Charged Off"
## [2] "Current"
## [3] "Default"
## [4] "Does not meet the credit policy. Status:Charged Off"
## [5] "Does not meet the credit policy. Status:Fully Paid"
## [6] "Fully Paid"
## [7] "In Grace Period"
## [8] "Late (16-30 days)"
## [9] "Late (31-120 days)"
```

```
trainv2$loan_status <- factor(trainv2$loan_status)
levels(trainv2$loan_status)
```

```
## [1] "Charged Off" "Current"     "Fully Paid"
```

```
testv2$loan_status <- factor(testv2$loan_status)
```

Step 3 – Training models

Null Model

Proportions of loan status for train and test datasets can be seen below. Fully paid had proportions between 80% and 82% for samples. We have to consider dominance of this property on machine learning algorithms.

```
trainv2 %>%
  group_by(loan_status) %>%
  summarise(n = n()) %>%
  mutate(freq = n/sum(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 3
##   loan_status     n   freq
##   <fct>       <int>  <dbl>
## 1 Charged Off  1677 0.168
## 2 Current       303 0.0303
## 3 Fully Paid   8020 0.802
```

```
testv2 %>%
  group_by(loan_status) %>%
  summarise(n = n()) %>%
  mutate(freq = n/sum(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 3
##   loan_status     n   freq
##   <fct>       <int>  <dbl>
## 1 Charged Off   398 0.159
## 2 Current        73 0.0292
## 3 Fully Paid   2029 0.812
```

kNN

kNN method was implemented after required normalization. I chose k as 4 firstly. The accuracies were 85.88% and 77.68% for train and test data respectively. Detailed proportions can be seen on CrossTable. Over-fitting is on the acceptable range for kNN method.

```r
# Normalization function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

# Prepare data
train_knnv2 <- trainv2 %>% mutate_if(is.factor, as.numeric)
test_knnv2 <- testv2 %>% mutate_if(is.factor, as.numeric)

# Normalization
train_knnv2n  <- as.data.frame(lapply(train_knnv2[2:11], normalize))
test_knnv2n  <- as.data.frame(lapply(test_knnv2[2:11], normalize))

# Prediction for train data
pred_knntrainv2 <- knn(train_knnv2n, train_knnv2n, cl= train_knnv2$loan_status, k=4)

# Evaluating model performance
CrossTable(x = train_knnv2$loan_status, y = pred_knntrainv2,
           prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  10000
##
##
##                         | pred_knntrainv2
## train_knnv2$loan_status |         1 |         2 |         3 | Row Total |
## ------------------------|-----------|-----------|-----------|-----------|
##                       1 |       949 |        28 |       700 |      1677 |
##                         |     0.566 |     0.017 |     0.417 |     0.168 |
##                         |     0.664 |     0.230 |     0.083 |           |
##                         |     0.095 |     0.003 |     0.070 |           |
## ------------------------|-----------|-----------|-----------|-----------|
##                       2 |        95 |        61 |       147 |       303 |
##                         |     0.314 |     0.201 |     0.485 |     0.030 |
##                         |     0.066 |     0.500 |     0.017 |           |
##                         |     0.009 |     0.006 |     0.015 |           |
## ------------------------|-----------|-----------|-----------|-----------|
##                       3 |       386 |        33 |      7601 |      8020 |
##                         |     0.048 |     0.004 |     0.948 |     0.802 |
```

```
##                          |     0.270 |     0.270 |     0.900 |           |
##                          |     0.039 |     0.003 |     0.760 |           |
## -------------------------|-----------|-----------|-----------|-----------|
##             Column Total |      1430 |       122 |      8448 |     10000 |
##                          |     0.143 |     0.012 |     0.845 |           |
## -------------------------|-----------|-----------|-----------|-----------|
##
##
```

```r
mean(train_knnv2$loan_status == pred_knntrainv2)
```

```
## [1] 0.8611
```

```r
# Prediction for test data
pred_knntestv2 <- knn(train_knnv2n, test_knnv2n, cl= train_knnv2$loan_status, k=4)

# Evaluating model performance
CrossTable(x = test_knnv2$loan_status, y = pred_knntestv2,
           prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  2500
##
##
##                         | pred_knntestv2
## test_knnv2$loan_status |         1 |         2 |         3 | Row Total |
## ------------------------|-----------|-----------|-----------|-----------|
##                       1 |       146 |        10 |       242 |       398 |
##                         |     0.367 |     0.025 |     0.608 |     0.159 |
##                         |     0.379 |     0.370 |     0.116 |           |
##                         |     0.058 |     0.004 |     0.097 |           |
## ------------------------|-----------|-----------|-----------|-----------|
##                       2 |        40 |         2 |        31 |        73 |
##                         |     0.548 |     0.027 |     0.425 |     0.029 |
##                         |     0.104 |     0.074 |     0.015 |           |
##                         |     0.016 |     0.001 |     0.012 |           |
## ------------------------|-----------|-----------|-----------|-----------|
##                       3 |       199 |        15 |      1815 |      2029 |
##                         |     0.098 |     0.007 |     0.895 |     0.812 |
##                         |     0.517 |     0.556 |     0.869 |           |
##                         |     0.080 |     0.006 |     0.726 |           |
## ------------------------|-----------|-----------|-----------|-----------|
##            Column Total |       385 |        27 |      2088 |      2500 |
##                         |     0.154 |     0.011 |     0.835 |           |
## ------------------------|-----------|-----------|-----------|-----------|
```

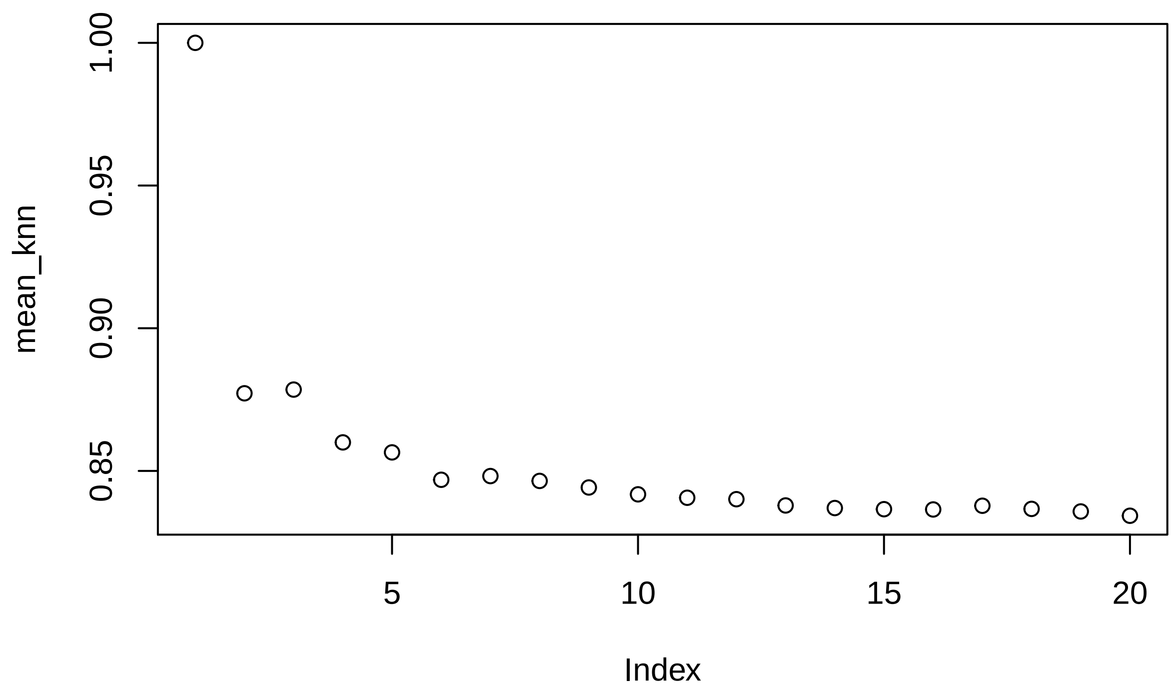
```
##
##
```

```r
mean(test_knnv2$loan_status == pred_knntestv2)
```

```
## [1] 0.7852
```

This time, I tried to find effect of different k values via a for loop. I excluded CrossTable this time for ease of readability. Plots of accuracies can be seen below. Train data had perfect fit when k is equal to 1 while test data had lowest accuracy rate. With increasing k value, accuracy of train data decreases and accuracy of test data increases. On the other hand, we should consider that increasing k value may cause to domination by fully paid results while low value of k cause augmented local influence. So, k should be between 3 and 4 to get more accurate predictions while we can arrange any other k up to cost of decisions. Decision maker should consider total profit for his case.

```r
# Normalization function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}


tic()

# Prepare data
mean_knn <- c()
train_knnv2 <- trainv2 %>%
  mutate_if(is.factor, as.numeric) %>%
  select(loan_status, everything())

test_knnv2 <- testv2 %>%
  mutate_if(is.factor, as.numeric) %>%
  select(loan_status, everything())

# Normalization
train_knnv2n  <- as.data.frame(lapply(train_knnv2[2:11], normalize))
test_knnv2n  <- as.data.frame(lapply(test_knnv2[2:11], normalize))


# Loop for train data
for (k in c(1:20)) {
pred_knntrainv2 <- knn(train_knnv2n, train_knnv2n, cl= train_knnv2$loan_status, k=k)

mean_knn[k] <- mean(train_knnv2$loan_status == pred_knntrainv2)

}
# Evaluating model performance
plot(mean_knn)
```



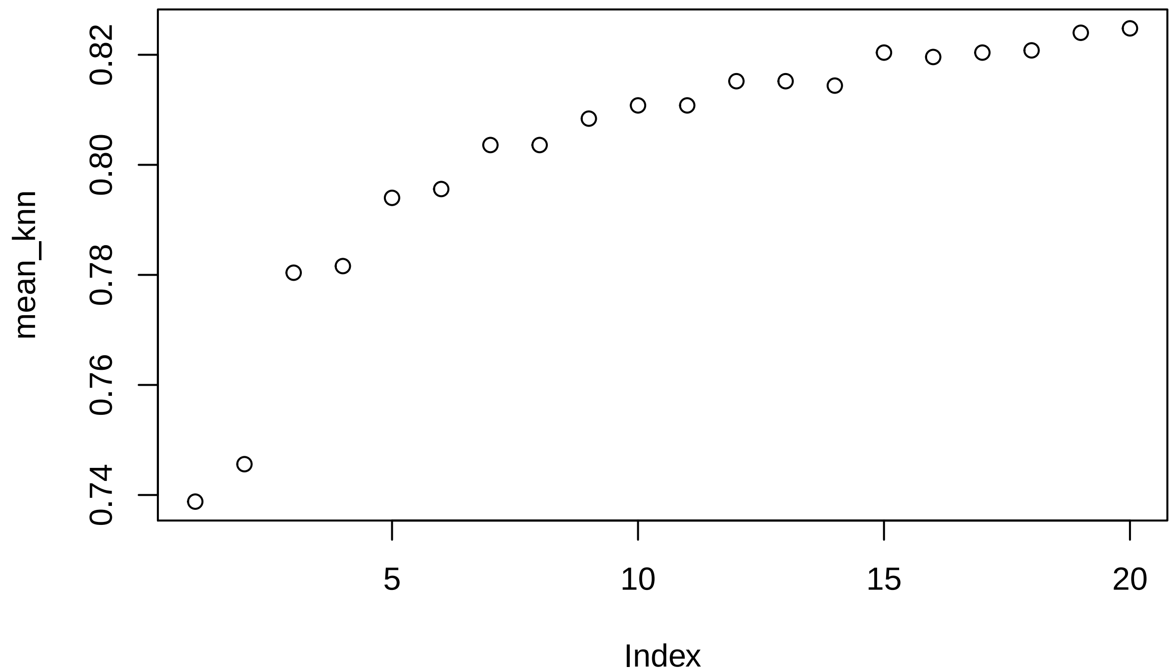

```
mean_knn
```

```
##  [1] 1.0000 0.8772 0.8785 0.8600 0.8565 0.8469 0.8482 0.8465 0.8442 0.8418
## [11] 0.8406 0.8401 0.8379 0.8370 0.8366 0.8365 0.8378 0.8367 0.8358 0.8343
```

```
toc()
```

```
## 17.59 sec elapsed
```

```
# Loop for test data
for (k in c(1:20)) {
pred_knntestv2 <- knn(train_knnv2n, test_knnv2n, cl= train_knnv2$loan_status, k=k)

mean_knn[k] <- mean(test_knnv2$loan_status == pred_knntestv2)

}
# Evaluating model performance
plot(mean_knn)
```




```r
mean_knn
```

```
##  [1] 0.7388 0.7456 0.7804 0.7816 0.7940 0.7956 0.8036 0.8036 0.8084 0.8108
## [11] 0.8108 0.8152 0.8152 0.8144 0.8204 0.8196 0.8204 0.8208 0.8240 0.8248
```

```r
toc()
```

Boosted C5.0

One of the best way to learn loans is classification trees. Default decision tree via C5.0 can be seen below.

```r
train_c50 <- trainv2 %>% select(-emp_length)
test_c50 <- testv2 %>% select(-emp_length)

modelc50 <- C5.0(train_c50[,-1], train_c50$loan_status)

modelc50
```

```
##
## Call:
## C5.0.default(x = train_c50[, -1], y = train_c50$loan_status)
##
## Classification Tree
## Number of samples: 10000
## Number of predictors: 9
##
## Tree size: 118
##
```

```
## Non-standard options: attempt to group attributes
```

```r
summary(modelc50)
```

```
##
## Call:
## C5.0.default(x = train_c50[, -1], y = train_c50$loan_status)
##
##
## C5.0 [Release 2.07 GPL Edition]      Fri Nov 27 00:32:27 2020
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 10000 cases (10 attributes) from undefined.data
##
## Decision tree:
##
## last_pymnt_amnt > 1309.69:
## :...last_pymnt_amnt > 2135: Fully Paid (4469/5)
## :   last_pymnt_amnt <= 2135:
## :   :...funded_amnt <= 26150: Fully Paid (421/7)
## :       funded_amnt > 26150:
## :       :...mo_sin_old_rev_tl_op > 173: Fully Paid (15/1)
## :           mo_sin_old_rev_tl_op <= 173:
## :           :...acc_open_past_24mths <= 7: Charged Off (10/2)
## :               acc_open_past_24mths > 7: Fully Paid (2)
## last_pymnt_amnt <= 1309.69:
## :...int_rate > 14.47:
##     :...funded_amnt <= 9900:
##     :   :...last_pymnt_amnt > 594.34: Fully Paid (73/1)
##     :   :   last_pymnt_amnt <= 594.34:
##     :   :   :...last_pymnt_amnt <= 38.72: Fully Paid (53/6)
##     :   :       last_pymnt_amnt > 38.72:
##     :   :       :...num_bc_sats > 7: Fully Paid (47/8)
##     :   :           num_bc_sats <= 7:
##     :   :           :...acc_open_past_24mths > 6:
##     :   :               :...last_pymnt_amnt > 385.79: Fully Paid (4)
##     :   :               :   last_pymnt_amnt <= 385.79:
##     :   :               :   :...avg_cur_bal > 9260: Fully Paid (21/8)
##     :   :               :       avg_cur_bal <= 9260:
##     :   :               :       :...last_pymnt_amnt <= 152.23: Fully Paid (12/5)
##     :   :               :           last_pymnt_amnt > 152.23: Charged Off (47/8)
##     :   :               acc_open_past_24mths <= 6:
##     :   :               :...int_rate <= 17.27: Fully Paid (240/74)
##     :   :                   int_rate > 17.27:
##     :   :                   :...funded_amnt <= 3625: Fully Paid (64/19)
##     :   :                       funded_amnt > 3625:
##     :   :                       :...last_pymnt_amnt > 246.77:
##     :   :                           :...num_bc_sats <= 3: Fully Paid (47/16)
##     :   :                           :   num_bc_sats > 3: Charged Off (35/15)
##     :   :                           last_pymnt_amnt <= 246.77:
##     :   :                           :...funded_amnt > 7300:
##     :   :                               :...avg_cur_bal <= 3796: Current (2)
##     :   :                               :   avg_cur_bal > 3796: Charged Off (3)
```

```
##     :   :                               funded_amnt <= 7300:
##     :   :                               :...annual_inc > 112214: Fully Paid (4)
##     :   :                                   annual_inc <= 112214: [S1]
##     :   funded_amnt > 9900:
##     :   :...last_pymnt_amnt <= 199.97:
##     :       :...last_pymnt_amnt <= 24.85:
##     :       :   :...last_pymnt_amnt <= 0: Charged Off (3)
##     :       :   :   last_pymnt_amnt > 0: Fully Paid (63/3)
##     :       :   last_pymnt_amnt > 24.85:
##     :       :   :...last_pymnt_amnt <= 102.08:
##     :       :       :...last_pymnt_amnt <= 94.18: Fully Paid (36/14)
##     :       :       :   last_pymnt_amnt > 94.18: Charged Off (16)
##     :       :       last_pymnt_amnt > 102.08:
##     :       :       :...last_pymnt_amnt <= 147.31: Fully Paid (15)
##     :       :           last_pymnt_amnt > 147.31:
##     :       :           :...last_pymnt_amnt <= 158.61: Charged Off (4)
##     :       :               last_pymnt_amnt > 158.61: Fully Paid (7/1)
##     :       last_pymnt_amnt > 199.97:
##     :       :...last_pymnt_amnt > 972.29:
##     :           :...int_rate <= 16.29: Fully Paid (48/8)
##     :           :   int_rate > 16.29: Charged Off (82/35)
##     :           last_pymnt_amnt <= 972.29:
##     :           :...int_rate <= 18.85:
##     :               :...last_pymnt_amnt <= 333.7:
##     :               :   :...int_rate > 16.24:
##     :               :   :   :...annual_inc <= 95000: Charged Off (66/21)
##     :               :   :   :   annual_inc > 95000: Current (4/1)
##     :               :   :   int_rate <= 16.24:
##     :               :   :   :...int_rate > 15.61: Fully Paid (4/1)
##     :               :   :       int_rate <= 15.61: [S2]
##     :               :   last_pymnt_amnt > 333.7:
##     :               :   :...funded_amnt <= 12975: Fully Paid (151/58)
##     :               :       funded_amnt > 12975:
##     :               :       :...last_pymnt_amnt <= 499.67: Charged Off (229/102)
##     :               :           last_pymnt_amnt > 499.67:
##     :               :           :...funded_amnt <= 20050: Fully Paid (124/52)
##     :               :               funded_amnt > 20050: [S3]
##     :               int_rate > 18.85:
##     :               :...mort_acc <= 1:
##     :                   :...acc_open_past_24mths > 2: Charged Off (236/62)
##     :                   :   acc_open_past_24mths <= 2:
##     :                   :   :...acc_open_past_24mths <= 1: Charged Off (27/9)
##     :                   :       acc_open_past_24mths > 1: [S4]
##     :                   mort_acc > 1:
##     :                   :...acc_open_past_24mths > 11: Charged Off (8)
##     :                       acc_open_past_24mths <= 11:
##     :                       :...funded_amnt <= 11600: Charged Off (15/6)
##     :                           funded_amnt > 11600:
##     :                           :...avg_cur_bal <= 9130: [S5]
##     :                               avg_cur_bal > 9130:
##     :                               :...num_bc_sats <= 1: Charged Off (10/1)
##     :                                   num_bc_sats > 1:
##     :                                   :...num_bc_sats > 4: [S6]
##     :                                       num_bc_sats <= 4:
```

```
##     :                                       :...int_rate <= 19.52: [S7]
##     :                                           int_rate > 19.52:
##     :                                           :...mort_acc > 3: [S8]
##     :                                               mort_acc <= 3: [S9]
##     int_rate <= 14.47:
##     :...int_rate <= 7.62: Fully Paid (318/13)
##         int_rate > 7.62:
##         :...last_pymnt_amnt <= 66.45: Fully Paid (169/5)
##             last_pymnt_amnt > 66.45:
##             :...last_pymnt_amnt > 806.93: Fully Paid (271/28)
##                 last_pymnt_amnt <= 806.93:
##                 :...funded_amnt <= 9975:
##                     :...last_pymnt_amnt > 332.1: Fully Paid (101/5)
##                     :   last_pymnt_amnt <= 332.1:
##                     :   :...mo_sin_old_rev_tl_op > 253: Fully Paid (104/14)
##                     :       mo_sin_old_rev_tl_op <= 253:
##                     :       :...acc_open_past_24mths <= 3: Fully Paid (292/72)
##                     :           acc_open_past_24mths > 3:
##                     :           :...acc_open_past_24mths > 9:
##                     :               :...int_rate <= 9.25: Fully Paid (2)
##                     :               :   int_rate > 9.25: Charged Off (11/1)
##                     :               acc_open_past_24mths <= 9:
##                     :               :...annual_inc > 37500: Fully Paid (188/54)
##                     :                   annual_inc <= 37500:
##                     :                   :...int_rate > 14.16: Fully Paid (6)
##                     :                       int_rate <= 14.16: [S10]
##                     funded_amnt > 9975:
##                     :...funded_amnt > 21350:
##                         :...last_pymnt_amnt <= 713.22:
##                         :   :...last_pymnt_amnt <= 457.31: [S11]
##                         :   :   last_pymnt_amnt > 457.31:
##                         :   :   :...last_pymnt_amnt <= 690.12: Current (73/39)
##                         :   :       last_pymnt_amnt > 690.12: Charged Off (7/1)
##                         :   last_pymnt_amnt > 713.22:
##                         :   :...funded_amnt <= 32500: Fully Paid (58/9)
##                         :       funded_amnt > 32500:
##                         :       :...acc_open_past_24mths <= 8: Current (9/1)
##                         :           acc_open_past_24mths > 8: Fully Paid (2)
##                         funded_amnt <= 21350:
##                         :...last_pymnt_amnt > 465.62: Fully Paid (461/109)
##                             last_pymnt_amnt <= 465.62:
##                             :...funded_amnt > 14500:
##                                 :...last_pymnt_amnt <= 275.2: Fully Paid (19/3)
##                                 :   last_pymnt_amnt > 275.2:
##                                 :   :...num_bc_sats > 10: Charged Off (5/1)
##                                 :       num_bc_sats <= 10:
##                                 :       :...int_rate <= 11.67:
##                                 :           :...funded_amnt > 19050: Current (11/3)
##                                 :           :   funded_amnt <= 19050: [S12]
##                                 :           int_rate > 11.67: [S13]
##                                 funded_amnt <= 14500:
##                                 :...last_pymnt_amnt > 310.49: Fully Paid (443/134)
##                                     last_pymnt_amnt <= 310.49:
##                                     :...int_rate > 13.98: Charged Off (8/1)
```

```
##                                         int_rate <= 13.98:
##                                         :...last_pymnt_amnt <= 205.86: [S14]
##                                             last_pymnt_amnt > 205.86:
##                                             :...int_rate <= 11.14: [S15]
##                                                 int_rate > 11.14: [S16]
##
## SubTree [S1]
##
## mo_sin_old_rev_tl_op <= 97: Charged Off (25/2)
## mo_sin_old_rev_tl_op > 97:
## :...annual_inc <= 39000: Fully Paid (19/7)
##     annual_inc > 39000:
##     :...mo_sin_old_rev_tl_op <= 106: Fully Paid (3)
##         mo_sin_old_rev_tl_op > 106: Charged Off (22/2)
##
## SubTree [S2]
##
## acc_open_past_24mths > 2: Charged Off (36/10)
## acc_open_past_24mths <= 2:
## :...mo_sin_old_rev_tl_op <= 239: Fully Paid (12/5)
##     mo_sin_old_rev_tl_op > 239: Current (2)
##
## SubTree [S3]
##
## acc_open_past_24mths > 4: Charged Off (76/26)
## acc_open_past_24mths <= 4:
## :...last_pymnt_amnt <= 627.06:
##     :...funded_amnt <= 30700: Current (46/26)
##     :   funded_amnt > 30700: Fully Paid (3)
##     last_pymnt_amnt > 627.06:
##     :...mort_acc > 7: Fully Paid (3)
##         mort_acc <= 7:
##         :...funded_amnt > 29875: Charged Off (38/18)
##             funded_amnt <= 29875:
##             :...mo_sin_old_rev_tl_op > 238: Fully Paid (5)
##                 mo_sin_old_rev_tl_op <= 238:
##                 :...funded_amnt > 26525: Charged Off (9)
##                     funded_amnt <= 26525:
##                     :...mort_acc <= 2: Charged Off (13/3)
##                         mort_acc > 2: Fully Paid (7/1)
##
## SubTree [S4]
##
## mo_sin_old_rev_tl_op > 322: Charged Off (2/1)
## mo_sin_old_rev_tl_op <= 322:
## :...int_rate <= 23.83: Fully Paid (27/10)
##     int_rate > 23.83: Charged Off (5)
##
## SubTree [S5]
##
## acc_open_past_24mths > 4: Charged Off (30/6)
## acc_open_past_24mths <= 4:
## :...funded_amnt <= 31575: Current (11/3)
##     funded_amnt > 31575: Charged Off (4)
```

```
##
## SubTree [S6]
##
## annual_inc <= 131053: Charged Off (48/15)
## annual_inc > 131053:
## :...last_pymnt_amnt <= 860.41: Fully Paid (2)
##     last_pymnt_amnt > 860.41: Current (2)
##
## SubTree [S7]
##
## funded_amnt > 27700: Charged Off (7)
## funded_amnt <= 27700:
## :...avg_cur_bal > 21114: Current (5)
##     avg_cur_bal <= 21114:
##     :...avg_cur_bal <= 11707: Current (2)
##         avg_cur_bal > 11707: Charged Off (5)
##
## SubTree [S8]
##
## num_bc_sats > 3: Current (3)
## num_bc_sats <= 3:
## :...acc_open_past_24mths <= 6: Charged Off (9/5)
##     acc_open_past_24mths > 6: Current (5/1)
##
## SubTree [S9]
##
## num_bc_sats <= 2: Charged Off (8)
## num_bc_sats > 2:
## :...int_rate <= 21.7: Fully Paid (11/4)
##     int_rate > 21.7:
##     :...annual_inc <= 74500: Current (2)
##         annual_inc > 74500: Charged Off (4)
##
## SubTree [S10]
##
## acc_open_past_24mths <= 8: Charged Off (56/23)
## acc_open_past_24mths > 8: Fully Paid (2)
##
## SubTree [S11]
##
## mo_sin_old_rev_tl_op <= 343: Fully Paid (16)
## mo_sin_old_rev_tl_op > 343: Charged Off (2)
##
## SubTree [S12]
##
## last_pymnt_amnt <= 353.25: Current (13/4)
## last_pymnt_amnt > 353.25: Fully Paid (11/4)
##
## SubTree [S13]
##
## mort_acc > 1: Charged Off (27/13)
## mort_acc <= 1:
## :...acc_open_past_24mths <= 1: Charged Off (5)
##     acc_open_past_24mths > 1: Current (19/6)
```

```
##
## SubTree [S14]
##
## num_bc_sats <= 1: Charged Off (2)
## num_bc_sats > 1: Fully Paid (9)
##
## SubTree [S15]
##
## mo_sin_old_rev_tl_op <= 134: Current (4)
## mo_sin_old_rev_tl_op > 134:
## :...mo_sin_old_rev_tl_op <= 172: Fully Paid (6)
##     mo_sin_old_rev_tl_op > 172:
##     :...last_pymnt_amnt <= 241.7: Current (3)
##         last_pymnt_amnt > 241.7: Fully Paid (3/1)
##
## SubTree [S16]
##
## funded_amnt > 14150: Fully Paid (2)
## funded_amnt <= 14150:
## :...int_rate > 13.11:
##     :...num_bc_sats <= 3: Current (3)
##     :   num_bc_sats > 3: Charged Off (9/1)
##     int_rate <= 13.11:
##     :...funded_amnt > 11300: Current (11/1)
##         funded_amnt <= 11300:
##         :...acc_open_past_24mths > 4: Charged Off (3)
##             acc_open_past_24mths <= 4:
##             :...num_bc_sats <= 2: Charged Off (2)
##                 num_bc_sats > 2: Current (6/2)
##
##
## Evaluation on training data (10000 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##     118 1242(12.4%)   <<
##
##
##     (a)   (b)   (c)    <-classified as
##    ----  ----  ----
##     879    48   750    (a): class Charged Off
##     139   149    15    (b): class Current
##     251    39  7730    (c): class Fully Paid
##
##
##  Attribute usage:
##
##  100.00% last_pymnt_amnt
##   50.83% int_rate
##   47.73% funded_amnt
##   19.01% acc_open_past_24mths
##    8.73% mo_sin_old_rev_tl_op
```

```
##    8.40% num_bc_sats
##    6.14% mort_acc
##    4.53% annual_inc
##    2.53% avg_cur_bal
##
##
## Time: 0.1 secs
```

```
fittedc50train <- predict(modelc50, newdata = train_c50)

print(paste('Accuracy for train:', mean(fittedc50train == trainv2$loan_status)))
```

```
## [1] "Accuracy for train: 0.8758"
```

```
# test

fittedc50test <- predict(modelc50, newdata = test_c50)

print(paste('Accuracy for test:', mean(fittedc50test == testv2$loan_status)))
```

```
## [1] "Accuracy for test: 0.8336"
```

C5.0 could be developed by boosting. I excluded summary of the new model because of ease of readability.

```
## Boosting the accuracy of decision trees
# boosted decision tree with 10 trials

modelc50boosted <- C5.0(train_c50[,-1], train_c50$loan_status, trials = 10)

modelc50boosted
```

```
##
## Call:
## C5.0.default(x = train_c50[, -1], y = train_c50$loan_status, trials = 10)
##
## Classification Tree
## Number of samples: 10000
## Number of predictors: 9
##
## Number of boosting iterations: 10
## Average tree size: 112.5
##
## Non-standard options: attempt to group attributes
```

```
#summary(modelc50boosted)
```

```
fittedc50trainboosted <- predict(modelc50boosted, newdata = train_c50)

print(paste('Accuracy for boosted train data:', mean(fittedc50trainboosted == trainv2$loan_status)))
```

```
## [1] "Accuracy for boosted train data: 0.9217"
```

```
# test

fittedc50testboosted <- predict(modelc50boosted, newdata = test_c50)

print(paste('Accuracy for boosted test data:', mean(fittedc50testboosted == testv2$loan_status)))
```

```
## [1] "Accuracy for boosted test data: 0.8308"
```

Random Forest

Accuracy is perfect for the train dataset. accuracy is 84.64% for test data.

```r
tic()
rf <- randomForest(loan_status ~ ., data = trainv2)

rf
```

```
##
## Call:
##  randomForest(formula = loan_status ~ ., data = trainv2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 15.9%
## Confusion matrix:
##             Charged Off Current Fully Paid class.error
## Charged Off         774       3        900  0.53846154
## Current             180       3        120  0.99009901
## Fully Paid          386       1       7633  0.04825436
```

```r
summary(rf)
```

```
##                 Length Class  Mode
## call                3  -none- call
## type                1  -none- character
## predicted       10000  factor numeric
## err.rate         2000  -none- numeric
## confusion          12  -none- numeric
## votes           30000  matrix numeric
## oob.times       10000  -none- numeric
## classes             3  -none- character
## importance         10  -none- numeric
## importanceSD        0  -none- NULL
## localImportance     0  -none- NULL
## proximity           0  -none- NULL
## ntree               1  -none- numeric
## mtry                1  -none- numeric
## forest             14  -none- list
## y               10000  factor numeric
## test                0  -none- NULL
## inbag               0  -none- NULL
## terms               3  terms  call
```

```r
rf_predtrain <- predict(rf, trainv2)
confusionMatrix(data=rf_predtrain, trainv2$loan_status)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Charged Off Current Fully Paid
##   Charged Off        1677       0          0
##   Current               0     303          0
##   Fully Paid            0       0       8020
##
```

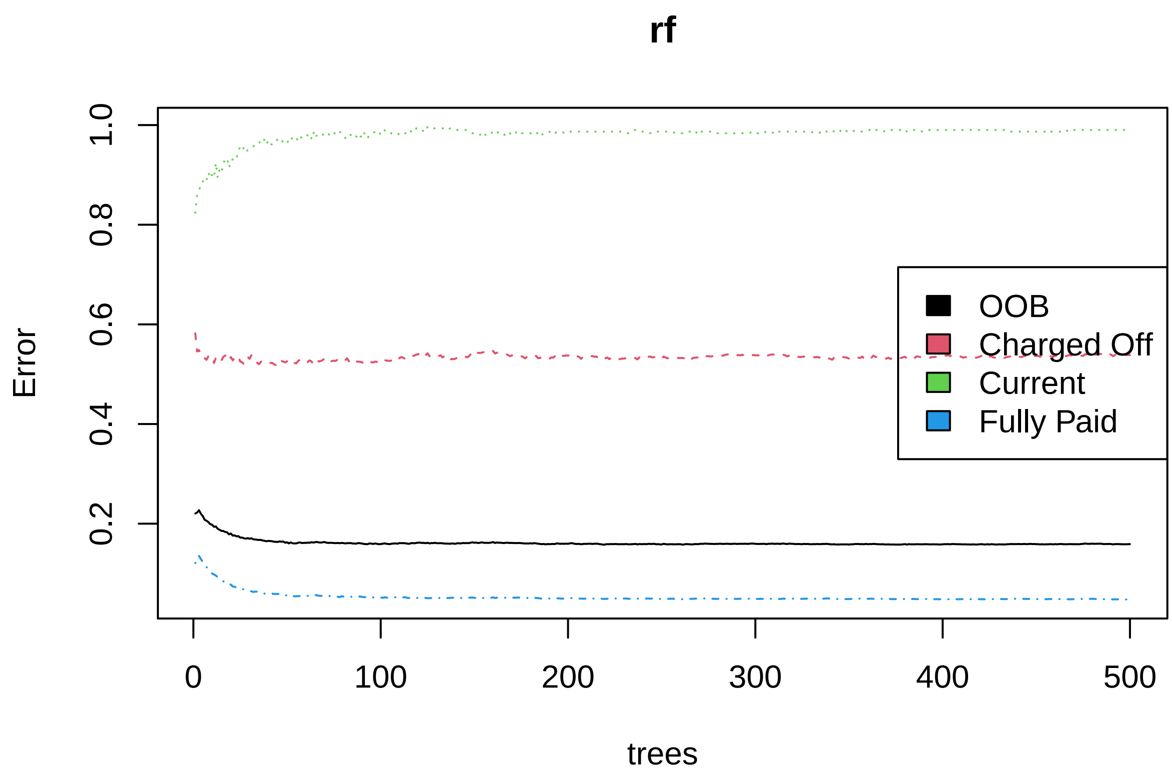
```
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9996, 1)
##     No Information Rate : 0.802
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Charged Off Class: Current Class: Fully Paid
## Sensitivity                      1.0000         1.0000             1.000
## Specificity                      1.0000         1.0000             1.000
## Pos Pred Value                   1.0000         1.0000             1.000
## Neg Pred Value                   1.0000         1.0000             1.000
## Prevalence                       0.1677         0.0303             0.802
## Detection Rate                   0.1677         0.0303             0.802
## Detection Prevalence             0.1677         0.0303             0.802
## Balanced Accuracy                1.0000         1.0000             1.000
```

```r
rf_predtest <- predict(rf, testv2)
confusionMatrix(data=rf_predtest, testv2$loan_status)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Charged Off Current Fully Paid
##   Charged Off         191      40        105
##   Current               1       1          0
##   Fully Paid          206      32       1924
##
## Overall Statistics
##
##                Accuracy : 0.8464
##                  95% CI : (0.8317, 0.8603)
##     No Information Rate : 0.8116
##     P-Value [Acc > NIR] : 2.913e-06
##
##                   Kappa : 0.4449
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Charged Off Class: Current Class: Fully Paid
## Sensitivity                      0.4799         0.0137            0.9483
## Specificity                      0.9310         0.9996            0.4947
## Pos Pred Value                   0.5685         0.5000            0.8899
## Neg Pred Value                   0.9043         0.9712            0.6893
## Prevalence                       0.1592         0.0292            0.8116
## Detection Rate                   0.0764         0.0004            0.7696
## Detection Prevalence             0.1344         0.0008            0.8648
```

```
## Balanced Accuracy                    0.7055          0.5066              0.7215
```

```r
err <- rf$err.rate
oob_err <- err[nrow(err), "OOB"]
plot(rf)
legend(x = "right",
       legend = colnames(err),
       fill = 1:ncol(err))
```




```r
toc()
```

```
## 20.22 sec elapsed
```

Random Forest via ranger

Accuracy is perfect for the train dataset. Accuracy is 84.64% for test dataset again by ranger package.

```r
tic()
rfranger <- ranger(loan_status ~ ., data = trainv2, num.threads = 4 )

rfranger
```

```
## Ranger result
##
## Call:
##  ranger(loan_status ~ ., data = trainv2, num.threads = 4)
##
## Type:                             Classification
## Number of trees:                  500
```

```
## Sample size:                      10000
## Number of independent variables:  10
## Mtry:                             3
## Target node size:                 1
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error:             15.96 %
```

```
rfranger$confusion.matrix
```

```
##               predicted
## true           Charged Off Current Fully Paid
##   Charged Off          768       6        903
##   Current              174       4        125
##   Fully Paid           384       4       7632
```

```
summary(rfranger)
```

```
##                           Length Class         Mode
## predictions               10000  factor        numeric
## num.trees                     1  -none-        numeric
## num.independent.variables     1  -none-        numeric
## mtry                          1  -none-        numeric
## min.node.size                 1  -none-        numeric
## prediction.error              1  -none-        numeric
## forest                        9  ranger.forest list
## confusion.matrix              9  table         numeric
## splitrule                     1  -none-        character
## treetype                      1  -none-        character
## call                          4  -none-        call
## importance.mode               1  -none-        character
## num.samples                   1  -none-        numeric
## replace                       1  -none-        logical
```

```
rf_predrangertrain <- predict(rfranger, trainv2)

confusionMatrix(data=rf_predrangertrain$predictions, trainv2$loan_status)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Charged Off Current Fully Paid
##   Charged Off         1677       0          0
##   Current                0     303          0
##   Fully Paid             0       0       8020
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9996, 1)
##     No Information Rate : 0.802
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: Charged Off Class: Current Class: Fully Paid
## Sensitivity                      1.0000         1.0000             1.000
## Specificity                      1.0000         1.0000             1.000
## Pos Pred Value                   1.0000         1.0000             1.000
## Neg Pred Value                   1.0000         1.0000             1.000
## Prevalence                       0.1677         0.0303             0.802
## Detection Rate                   0.1677         0.0303             0.802
## Detection Prevalence             0.1677         0.0303             0.802
## Balanced Accuracy                1.0000         1.0000             1.000
```

```r
rf_predrangertest <- predict(rfranger, testv2)

confusionMatrix(data=rf_predrangertest$predictions, testv2$loan_status)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Charged Off Current Fully Paid
##   Charged Off         186      38        101
##   Current               1       3          0
##   Fully Paid          211      32       1928
##
## Overall Statistics
##
##                Accuracy : 0.8468
##                  95% CI : (0.8321, 0.8607)
##     No Information Rate : 0.8116
##     P-Value [Acc > NIR] : 2.254e-06
##
##                   Kappa : 0.4418
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Charged Off Class: Current Class: Fully Paid
## Sensitivity                      0.4673         0.0411            0.9502
## Specificity                      0.9339         0.9996            0.4841
## Pos Pred Value                   0.5723         0.7500            0.8881
## Neg Pred Value                   0.9025         0.9720            0.6930
## Prevalence                       0.1592         0.0292            0.8116
## Detection Rate                   0.0744         0.0012            0.7712
## Detection Prevalence             0.1300         0.0016            0.8684
## Balanced Accuracy                0.7006         0.5203            0.7171
```

```r
toc()
```

```
## 1.79 sec elapsed
```

Conclusion

3 different algorithms were implemented to the data. 1 alternative way and 1 boosting method were used to check and improve models. kNN method had least accurate results for test data while random forests had

best which is 84.64%. Ranger gave same results with randomForest package but execution time was smaller, 2.91 sec vs 18.99 sec respectively. C5.0 method results very close to random forest while boosting does not help to improve accuracy of outcomes.